AD-A104 939 TEXAS A AND M UNIV COLLEGE STATION INST OF STATISTICS F/G 12/1
MINIMUM MEAN SQUARE ERROR PREDICTION OF AUTOREGRESSIVE MOVING A--ETC(U)
APR 81 H J NEWTON, M PAGANO DAAG29-80-C-0070
UNCLASSIFIED TR-N-24 NL

1 OF 1
AD A104939

END
DATE FILMED
10 81
DTIC

TEXAS A&M UNIVERSITY

COLLEGE STATION, TEXAS 77843

INSTITUTE OF STATISTICS
Phone 713 845-3141

MINIMUM MEAN SQUARE ERROR PREDICTION OF AUTOREGRESSIVE MOVING AVERAGE TIME SERIES

by H. Joseph Newton and Marcello Pagano

Institute of Statistics
Texas A&M University

Statistical Science Division
State University of New York
at Buffalo

Technical Report No. N-24

April 1981

Texas A & M Research Foundation
Project No. 4226T

"Robust Statistical Data Analysis and Modeling"

Sponsored by the Office of Naval Research

Professor Emanuel Parzen, Principal Investigator

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(14) TR-N-24

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report N-24
2. GOVT ACCESSION NO.: AD-A104 939
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): (6) Minimum Mean Square Error Prediction of Autoregressive Moving Average Time Series.
5. TYPE OF REPORT & PERIOD COVERED: (9) Technical rept.
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): (10) H. Joseph Newton [redacted] Marcello Pagano
8. CONTRACT OR GRANT NUMBER(s): (15) DAAG29-80-C-0070; ONR N0001481MP10001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Texas A&M University, Institute of Statistics, College Station, TX 77843
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Research Office, P.O. Box 12277, Research Triangle Park, NC 27709
12. REPORT DATE: (11) Apr[redacted]81
13. NUMBER OF PAGES: 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): (12) 27
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distribution unlimited.
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): NA
18. SUPPLEMENTARY NOTES:
19. KEY WORDS (Continue on reverse side if necessary and identify by block number): Autoregressive Moving Average Time Series; Toeplitz matrix; modified Cholesky Decomposition
20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

A computer program is described and presented for calculating finite memory predictors and prediction variances for autoregressive moving average time series models. The Cholesky decomposition algorithm is used and a number of simplifying results are described and implemented in the program.

347380

MINIMUM MEAN SQUARE ERROR PREDICTION OF AUTOREGRESSIVE MOVING AVERAGE TIME SERIES

By H. Joseph Newton and Marcello Pagano

Institute of Statistics, Texas A&M University and Dept. of Biostatistics, Harvard University, U.S.A.

Keywords: Autoregressive Moving Average Time Series; Toeplitz Matrix; Modified Cholesky Decomposition

LANGUAGE

ISO Fortran

DESCRIPTION AND PURPOSE

Let Y(1), ..., Y(T) be a sample realization of a mixed autoregressive moving average time series $\{Y(t), t = 0, \pm 1, \ldots\}$ of order (p,q) (denoted ARMA(p,q)). Thus

$$\sum_{j=0}^{p} \alpha(j)Y(t-j) = \sum_{k=0}^{q} \beta(k)\varepsilon(t-k) \text{ , } t = 0, \pm 1, \ldots$$

for constants p,q , $\alpha(0) = \beta(0) = 1$, $\alpha(1), \ldots, \alpha(p)$, $\beta(1), \ldots, \beta(q)$, where $\varepsilon(\cdot)$ is a white noise series of zero mean uncorrelated random variables having variance σ^2. The zeros of the complex polynomials $g(z) = \sum_{j=0}^{p}\alpha(j)z^j$ and $h(z) = \sum_{k=0}^{q}\beta(k)z^k$ are assumed to be outside the unit circle.

Subroutine MXPD calculates exact, memory h, horizon t, minimum mean square error linear predictors $Y(t+h|t)$ and (optionally) prediction variances $\sigma^2_{t,h}$ of Y(t+h) given Y(1), ..., Y(t) for $h = h_1, \ldots, h_2$ and $t = t_1, \ldots, t_2$. Thus $Y(t+h|t)$ is that linear combination of Y(1), ..., Y(t) closest to Y(t+h) in the mean square sense and $\sigma^2_{t,h} = E\{Y(t+h) - Y(t+h|t)\}^2$ is the attained minimum mean square prediction variance.

If q = 0, Y is a pure autoregressive process of order p (AR(p)), while

if $p = 0$, Y is a pure moving average process of order q (MA(q)). Subroutines ARPD and MAPD calculate $Y(t+h|t)$ and (optionally) $\sigma^2_{t,h}$ for the AR(p) and MA(q) cases respectively. Separate subroutines are used for these cases due to significant simplifications in their algorithms from the general ARMA(p,q) case.

NUMERICAL METHOD

The method is based on numerous special properties of the modified Cholesky decomposition (MCD, see Wilkinson (1967)) of the autocovariance matrix of an ARMA(p,q) process. (See Newton and Pagano (1980) for details).

Let $[A]_{ij}$ denote the (i,j)th element of the matrix A and define $A_K \equiv \mathrm{TOEPL}\{a(0), \ldots, a(K-1)\}$ to be the $(K\times K)$ symmetric Toeplitz matrix having $[A_K]_{ij} = a(|i-j|)$. Let $A_K = L_K D_K L_K^T$, $K \geq 1$ be the MCD of the symmetric positive definite nested matrices $A_1, A_2, \ldots,$ i.e. $[A_{K+1}]_{ij} = [A_K]_{ij}$, for $i,j \leq K$, L_K is a unit lower triangular $(K\times K)$ matrix, and D_K is a diagonal matrix. Then the sequences of matrices L_K, D_K, L_K^{-1}, and D_K^{-1} are also nested and we write (for example) the (i,j)th element of any L_K for $K \geq \max(i,j)$ as $[L]_{ij}$.

Thus $[D]_{11} = [A]_{11}$,

$$[L]_{ij} = \{[A]_{ij} - \sum_{\ell=1}^{j-1} [L]_{i\ell}[D]_{\ell\ell}[L]_{j\ell}\}/[D]_{jj} \quad ,$$

$$[D]_{ii} = [A]_{ii} - \sum_{\ell=1}^{i-1} [D]_{\ell\ell}[L]^2_{i\ell} \; , \; 1 \leq j < i \leq K \; .$$

We define the following quantities for $K \geq 1$

a) $\Gamma_{Z,K} = \mathrm{TOEPL}\{R_Z(0), \ldots, R_Z(K-1)\} = L_{Z,K} D_{Z,K} L^T_{Z,K}$ where $R_Z(v) = E\{Z(t)Z(t+v)\}$, $v = 0, \pm 1, \ldots$ and $Z(\cdot)$ is an AR(p)

process having coefficients $\alpha(1), \ldots, \alpha(p)$ and white noise variance σ^2.

b) $\underset{\sim}{X}_K = (X(1), \ldots, X(K))^T = L_{Z,K}^{-1} \underset{\sim}{Y}_K$, where

$\underset{\sim}{Y}_K^T = (Y(1), \ldots, Y(K))$.

c) $\Gamma_{X,K} = E\{\underset{\sim}{X}_K \underset{\sim}{X}_K^T\} = L_{X,K} D_{X,K} L_{X,K}^T$. Note that $\Gamma_{X,K} = L_{Z,K}^{-1} \Gamma_{Y,K} L_{Z,K}^{-T}$.

d) $\underset{\sim}{e}_K = (e(1), \ldots, e(K))^T = L_{X,K}^{-1} \underset{\sim}{X}_K$.

Then

$$Y(t+h|t) = X(t+h|t) - \sum_{j=1}^{p} \alpha(j) Y(t+h-j|t)$$

$$\sigma_{t,h}^2 = \sum_{k=0}^{h-1} [L_Z L_X]_{t+h,t+h-k}^2 [D_X]_{t+h-k,t+h-k}$$

$$= \sum_{k=0}^{h-1} \left\{ [D_X]_{t+h-k,t+h-k} \left[\sum_{\ell=t+h-k}^{t+h} [L_Z]_{t+h,\ell} [L_X]_{\ell,t+h-k} \right]^2 \right\}$$

where $Y(t+h-j|t) = Y(t+h-j)$ if $j \geq h$, and

$$X(t+h|t) = \begin{cases} \sum_{k=h}^{q} [L_X]_{t+h,t+h-k} e(t+h-k) & , h = 1, \ldots, q \\ 0 & , h > q \end{cases}$$

Thus to obtain $Y(t+h|t)$ and $\sigma_{t,h}^2$ for $h = h_1, \ldots, h_2$ and $t = t_1, \ldots, t_2$ we need L_{Z,t_2+h_2}, $L_{Z,t_2+h_2}^{-1}$, L_{X,t_2+h_2}, and D_{X,t_2+h_2}. Significant reductions in computing time and storage requirements in obtaining the elements of these matrices are afforded by noting:

NOTE 1 Computing $L_{Z,t_2+h_2}^{-1}$

The j^{th} row of $L_{Z,K}^{-1}$ is given by

$$\ell_j^T = \begin{cases} (1, \underset{\sim}{0}_{K-1}^T) & , j = 1 \\ (\alpha_{j-1}(j-1), \ldots, \alpha_{j-1}(1), 1, \underset{\sim}{0}_{K-j}^T) & , 2 \leq j \leq p \\ (\underset{\sim}{0}_{j-p-1}^T, \alpha(p), \ldots, \alpha(1), 1, \underset{\sim}{0}_{K-j}^T), & p+1 \leq j \leq K \end{cases} \quad (1)$$

where $\alpha_k(\ell) = \alpha(\ell)$ if $k \geq p$ and

$$\alpha_j(i) = \frac{\alpha_{j+1}(i) - \alpha_{j+1}(j+1)\alpha_{j+1}(j+1-i)}{1-\alpha^2_{j+1}(j+1)} \; ; \; i = 1, \ldots, j < p \, . \tag{2}$$

Thus there are only $p(p+1)/2$ distinct nonzero, nonone elements in $L^{-1}_{Z,K}$, $K \geq p + 1$ and $X(1) = Y(1)$ while

$$X(j) = \begin{cases} Y(j) + \sum_{\ell=1}^{j-1} \alpha_{j-1}(\ell) Y(j-\ell) & , \; j = 2, \ldots, p \\ Y(j) + \sum_{\ell=1}^{p} \alpha(\ell) Y(j-\ell) & , \; j > p \end{cases} \tag{3}$$

<u>NOTE 2</u> Computing L_{Z,t_2+h_2}

Let $\gamma(0) = 1$, $\gamma(1)$, $\gamma(2)$, ... be the coefficients of the MA(∞) representation of $Z(\cdot)$, <u>i.e</u>.

$$\gamma(j) = - \sum_{\ell=\max(0,j-p)}^{j-1} \alpha(j-\ell)\gamma(\ell) \; , \quad j \geq 1 \, . \tag{4}$$

Then

$$[L_Z]_{p+j,p+j-\ell} = \gamma(\ell) \qquad , \; 0 \leq \ell \leq j \leq K-p \tag{5}$$

$$[L_Z]_{p+j,k} = - \sum_{r=1}^{p} \alpha(r) [L_Z]_{p+j-r,\ell} \; , \; \ell = 1, \ldots, p-1, \; j \geq 1 \, . \tag{6}$$

$$L_{Z,p} = (L^{-1}_{Z,p})^{-1} \Rightarrow [L_Z]_{j,j-k} = - \sum_{r=j-k+1}^{j} [L_Z]_{ir} [L^{-1}_Z]_{r,j-k} \, , \; k<j\leq p \tag{7}$$

$$\lim_{K \to \infty} \sum_{j=1}^{p-1} [L_Z]^2_{K,j} = 0 \tag{8}$$

$$\lim_{K \to \infty} \sum_{j=1}^{K} [L_Z]^2_{K,j} = R_Z(0)/\sigma^2 \tag{9}$$

$$\sum_{k=0}^{\infty} \gamma^2(k) = R_Z(0)/\sigma^2 \tag{10}$$

$$R_Z(0) = \sigma^2 \Big/ \prod_{j=1}^{p} (1-\alpha_j^2(j)) \tag{11}$$

By (5) and (6) the distinct elements of L_{Z,t_2+h_2} are contained in its first p columns, the last of which is $(\underset{\sim}{0}_{p-1}^T, 1, \gamma(1), \ldots, \gamma(t_2+h_2-p-j))^T$.

Thus MXPD finds the elements of L_{Z,t_2+h_2} by:

i) finding $L_{Z,p} = (L_{Z,p}^{-1})^{-1}$, ii) calculating $\gamma(1), \ldots, \gamma(M_1-p)$ (by (4)) where M_1-p is the smallest integer such that

$$\left| \sum_{j=0}^{M_1-p} \gamma^2(j) - R_Z(0)/\sigma^2 \right| < \delta , \tag{12}$$

iii) use (6) to obtain remaining elements of rows $p+1, \ldots, M_1$ of first p-1 columns of L_{Z,t_2+h_2}. Note that (10) says that such an M_1 exists (which hopefully is much less than t_2+h_2) while (8) and (9) say that all further elements of the first p columns of L_{Z,t_2+h_2} are arbitrarily close to zero. Thus there are $M_1 p$ elements to calculate and store for L_{Z,t_2+h_2}.

NOTE 3 Computing L_{X,t_2+h_2} , D_{X,t_2+h_2}

To compute L_{X,t_2+h_2} and D_{X,t_2+h_2} we note that (defining $\alpha_0(0) = 1$)

$$[\Gamma_X]_{ij} = \begin{cases} \displaystyle\sum_{m=\max(1,j-p)}^{j} \alpha_{j-1}(j-m) \sum_{\ell=\max(1,i-p)}^{i} \alpha_{i-1}(i-\ell) R_Y(\ell-m), & i,j \geq 1 \\ R_X(|i-j|) & , i,j > p, \ |i-j| \leq q \\ 0 & , 1 \leq j \leq p, \ i > p, \text{ and } \\ & \quad i-j > q \text{ or if } \\ & \quad i,j > p \text{ and } |i-j| > q \end{cases} \tag{13}$$

$$R_X(v) = \sigma^2 \sum_{k=0}^{q-v} \beta(k)\beta(k+v) \ , \quad v = 0, \ldots, q \ . \tag{14}$$

Thus the distinct elements of Γ_{X,t_2+h_2} are $R_X(0), \ldots, R_X(q)$ and the elements in the first p + q rows and p columns. Further only $R_Y(0), \ldots, R_Y(p+q)$ are required to obtain Γ_{X,t_2+h_2}. These elements are obtained via subroutine MXCV by solving first for $v = 0, \ldots, \max(p,q)$

$$\sum_{j=0}^{p} \alpha(j) R_Y(v-j) = \sum_{k=v}^{q} \beta(k) R_{Y\varepsilon}(v-k) = 0 \ , \quad v > q \tag{15}$$

where $R_{Y\varepsilon}(v) \equiv E\{Y(t)\varepsilon(t+v)\} = \delta_v \sigma^2$ if $v \geq 0$, where δ is the Kronecker delta, and

$$R_{Y\varepsilon}(-v) = \beta(v)\sigma^2 - \sum_{j=1}^{\min(v,p)} \alpha(j) R_{Y\varepsilon}(-v+j) \ , \quad v = 1, \ldots, q \ . \tag{16}$$

Then $R_Y(\max(p,q)+1), \ldots, R_Y(p+q)$ are obtained by (15) for $v = \max(p,q)+1, \ldots, p+q$.

To obtain L_{X,t_2+h_2} and D_{X,t_2+h_2} we note that the pattern of zeros in L_{X,t_2+h_2} is the same as that in the lower triangle of Γ_{X,t_2+h_2} and that the required elements of $\Gamma_{X,p+q}$ can be calculated as needed (by (13)) without storing them.

Thus $L_{X,p+q}$ is calculated and stored in one matrix. To obtain the q nonzero nonone, elements of the rows of the rest of L_{X,t_2+h_2} we have:

$$\lim_{K\to\infty} [L_X]_{K,K-j} = \beta(j) \qquad , \ j = 1, \ldots, q$$

$$\lim_{K\to\infty} [D_X]_{K,K} = \sigma^2 \ .$$

Thus rows p + q + 1, ... are calculated and stored in a matrix having q columns until the elements of L_X, D_X have converged (at row M_2 say) , i.e.

$$|\,[L_X]_{M_2,j} - \beta(q-j+1)| < \delta, \ |\,[D_X]_{M_2,M_2} - \sigma^2| < \delta, \ j = M_2-q, \ldots, M_2-1 \tag{17}$$

Further, $e(1) = X(1)$ while

$$e(j) = \begin{cases} X(j) - \sum_{\ell=1}^{j-1} [L_X]_{j,\ell}\, e(\ell) & , \ j = 2, \ldots, p+q \\ X(j) - \sum_{\ell=j-q}^{j-1} [L_X]_{j,\ell}\, e(\ell) & , \ j = p+q+1, \ldots, M_2 \\ X(j) - \sum_{\ell=1}^{q} \beta(\ell)\, e(j-\ell) & , \ j > M_2 \end{cases} \tag{18}$$

Also, if the $\sigma^2_{t,h}$ are not to be calculated then L_Z need not be calculated.

NOTE 4 Convergence of $L_Z L_X$

One further simplification is given by

$$\lim_{k \to \infty} [L_Z L_X]_{K,K-j} = \beta_\infty(j) \quad , \quad j \geq 1$$

where the $\beta_\infty(\cdot)$ are the coefficients of the MA(∞) representation of the ARMA(p,q) process Y. Thus for any $t \geq \max(M_1, M_2)$ we have

$$\sigma^2_{t,h} = \sigma^2 \sum_{k=0}^{h-1} \beta^2_\infty(k) ,$$

while if $t + h \geq M_1$ or $t + h \geq M_2$, the "converged" values are used for elements of the (t+h)th rows of L_Z, L_X in the expressions for $Y(t+h|t)$ and $\sigma^2_{t,h}$.

NOTE 5 From these observations we have Basic Structure of MXPD

i) Check input parameters.

ii) Find $R_Y(0), \ldots, R_Y(p+q)$ by (15) and (16) via subroutine MXCV (stored in the constant RY0 and (p+q)-vector RY).

iii) Find $L^{-1}_{Z,p}$ and $R_Z(0)$ by (1), (2), and (11) (stored in ($p \times p$) matrix ALZI and constant RZ0).

iv) Find M_1 and $[L_{Z,M_1+p}]_{ij}$, $1 \le i \le M_1$, $1 \le j \le p$ by (7), (6), and (12) (stored in the integer MONE and $(M_1 \times p)$ matrix ALZ).

v) Find $R_X(0)$, ..., $R_X(q)$ by (14) via subroutine MACV (stored in constant RX0 and q-vector RX).

vi) Find $L_{X,p+q}$ and $D_{X,p+q}$ (stored in the $(p+q)\times(p+q)$ matrix ALX1 and in the first (p+q) elements of the M_2-vector DX).

vii) Find M_2 and $[L_X]_{ij}$, $[D_X]_{ij}$, $i = p+q+1, \ldots, M_2$, $j = i-q, \ldots, i-1$ (stored in the integer MTWO and the $(M_2 \times q)$ matrix ALX2 and the rest of the (M_2)-vector DX).

viii) Find $\underset{\sim}{X}_{t_2}$, $\underset{\sim}{e}_{t_2}$ by (3) and (18) (stored in the t_2-vectors X and E).

ix) Find $Y(t+h|t)$, for $h = h_1, \ldots, h_2$, $t = t_1, \ldots, t_2$ (stored in the $(t_2-t_1+1)(h_2-h_1+1)$-vectors YPD where $Y(t+h|t) = \text{YPD}((t-t_1)(h_2-h_1+1)+(h-h_1+1)))$.

x) Find (optionally) $\sigma^2_{t,h}$ which is stored like YPD in a $(t_2-t_1+1)(h_2-h_1+1)$-vector PVAR .

NOTE 6 Taking advantage of convergence

To take advantage of the convergence in L_Z, L_X, and D_X the user specifies an absolute convergence criterion δ and integers IROWS1, IROWS2 as the row DIMENSION of arrays ALZ and ALX2, DX respectively. Thus IROWS1 and IROWS2 must be chosen to exceed what can reasonably be expected to be M_1 and M_2 respectively or else a nonconvergence failure indicator will be returned in IFAULT. Of course if IROWS1 or IROWS2 are given the value t_2+h_2 then convergence need not be reached for the algorithm to finis properly. Setting IROWS1 or IROWS2 smaller than t_2+h_2 allows the possibility of obtaining predictors for long time series with a minimum amount of required storage.

Further, if the $\sigma^2_{t,h}$ are not to be calculated, the matrix ALZ is not needed and IROWS1 can be set equal to 1.

NOTE 7 Algorithm for ARPD

For $q = 0$ we have $\Gamma_{Z,K} = \Gamma_{Y,K}$, $\Gamma_{X,K} = D_{Z,K} \Rightarrow L_{X,K} = I_K$, $D_{X,K} = D_{Z,K}$. Thus

$$Y(t+h|t) = - \sum_{j=1}^{p} \alpha(j) Y(t+h-j|t)$$

$$\sigma^2_{t,h} = \sum_{k=0}^{h-1} [L_Z]^2_{t+h,t+h-k} [D_Z]_{t+h-k,t+h-k}$$

$$= \sigma^2 \sum_{k=0}^{h-1} \gamma^2(k) \quad \text{if } t > p .$$

NOTE 8 Algorithm for MAPD

For $p = 0$ we have $\Gamma_{Z,K} = I_K \Rightarrow L_{Z,K} = D_{Z,K} = I_K$ and $\Gamma_{X,K} =$ TOEPL$\{R_X(0), \ldots, R_X(q), 0, \ldots, 0\}$, (so that calculation of ALX1 is avoided), $L_{X,K}\, \underset{\sim}{e}_K = \underset{\sim}{Y}_K$. Thus

$$Y(t+h|t) = \begin{cases} \sum_{k=h}^{q} [L_X]_{t+h,t+h-k}\, e(t+h-k) & , t+h < M_2 \\ \sum_{k=h}^{q} \beta(k) e(t+h-k) & , t+h \geq M_2 \\ 0 & , h > q \end{cases}$$

$$\sigma^2_{t,h} = \begin{cases} \sum_{k=0}^{h-1} [L_X]^2_{t+h,t+h-k} [D_X]_{t+h-k,t+h-k} & , t < M_2 \\ \sigma^2 \sum_{k=0}^{h-1} \beta^2(k) & , t \geq M_2 \end{cases}$$

STRUCTURE

SUBROUTINE MXPD (NP, NQ, ALPHA, BETA, SIGSQ, Y, IOPT, NT1, NT2, NH1, NH2, NYPD, NPVAR, NPPNH2, IROWS1, IROWS2, IROWS3, DEL, RYE, RYEO, RY, RYO, IP, YWK, RZO, RX, RXO, ALZI, ALZ, ALX1, ALX2, DX, MONE, MTWO, X, E, YPD, PVAR, IFAULT).

Formal parameters

NP	Integer	input: order of AR part of model
NQ	Integer	input: order of MA part of model
ALPHA	Real Array (NP)	input: coefficients of AR part of model
BETA	Real Array (NQ)	input: coefficients of MA part of model
SIGSQ	Real	input: variance of white noise in model
Y	Real Array (NT2)	input: data vector
IOPT	Integer	input: option switch equal to: 1 if both predictors and variances to be calculated 0 if only predictors desired
NT1	Integer	input: t_1 (first memory)
NT2	Integer	input: t_2 (last memory)
NH1	Integer	input: h_1 (first horizon)
NH2	Integer	input: h_2 (last horizon)
NYPD	Integer.	input: (NT2-NT1+1)(NH2-NH1+1)
NPVAR	Integer	input: same as NYPD if IOPT = 1, ≥ 1 if IOPT=0.
NPPNH2	Integer	input: NP + NH2
IROWS1	Integer	input: row dimension of ALZ in calling program ($\geq$NP+2 if IOPT =1, ≥ 1 if IOPT=0) see note 6 above
IROWS2	Integer	input: row dimension of ALX2, DX in calling program ($\geq$NP+NQ+1) See note 6 above.
IROWS3	Integer	input: row dimension of RY, IP, ALZI, ALX1 in calling program ($\geq$NP+NQ)
DEL	Real	input: absolute convergence criterion (see (12) and (17)).
RYE	Real Array (NQ)	output: $R_{Y\varepsilon}(-1), \ldots, R_{Y\varepsilon}(-q)$
RYEO	Real	output: $R_{Y\varepsilon}(0)$
RY	Real Array (NP+NQ)	output: $R_Y(1), \ldots, R_Y(p+q)$
RYO	Real	output: $R_Y(0)$
IP	Integer Array (IROWS3)	work:
YWK	Real Array (NPPNH2)	work:
RZO	Real	output: $R_Z(0)$
RX	Real Array (NQ)	output: $R_X(1), \ldots, R_X(q)$
RXO	Real	output: $R_X(0)$
ALZI	Real Array(IROWS3,IROWS3)	output: $L_{Z,p}^{-1}$
ALZ	Real Array (IROWS1, NP)	output: if IOPT = 1, ALZ contains $[L_Z]_{ij}$, $j = 1, \ldots, p$, $i = 1, \ldots, M_1$, if IOPT = 0, ALZ is not used.

ALX1	Real Array (IROWS3,IROWS3)	output: $L_{X,p+q}$
ALX2	Real Array (IROWS2,NQ)	output: $[L_X]_{ij}$, $i = p+q+1, \ldots, M_2$, $j = i-q, \ldots, i-1$.
DX	Real Array (IROWS2)	output: $[D_X]_i$, $i = 1, \ldots, M_2$
MONE	Integer	output: M_1 (see note 6 above)
MTWO	Integer	output: M_2 (see note 6 above)
X	Real Array (NT2)	output: $X(1), \ldots, X(t_2)$
E	Real Array (NT2)	output: $e(1), \ldots, e(t_2)$
YPD	Real Array (NYPD)	output: $((Y(t+h\|t),\ h = h_1, \ldots, h_2)$, $t = t_1, \ldots, t_2)$
PVAR	Real Array (NPVAR)	output: if IOPT = 1, $((\sigma^2_{t,h},\ h = h_1, \ldots, h_2),\ t = t_1, \ldots, t_2)$ if IOPT = 0, PVAR not used.
IFAULT	Integer	output: failure indicator

Failure Indications

Value of IFAULT	Meaning
0	no failure
1	NP<1, NQ<1, or IOPT not 0 or 1
2	NT1≤NP+NQ or NT1>NT2
3	NH1<1 or NH1>NH2
4	NYPD<(NT2-NT1+1)(NH2-NH1+1) or NPVAR<1 or IOPT = 1 and NPVAR<(NT2-NT1+1) (NH2-NH1+1)
5	IROWS1<NP+2 and IOPT = 1 or IROWS1<1 or NPPNH2<NP+NH2
6	IROWS2<NP+NQ+1
7	IROWS3<NP+NQ
8	SIGSQ<0
9	Singular matrix in Subroutine MXCV
10	An $\alpha_j(j) > 1$ (see(2))
11	IOPT = 1, IROWS1<NT2+NH2-NP, and convergence not reached.
12	Nonpositive $[D_X]_{ii}$ encountered
13	IROWS2<NT2+NH2-NP-NQ and convergence not reached.

SUBROUTINE ARPD (NP, ALPHA, SIGSQ, Y, IOPT, NT1, NT2, NH1, NH2, NYPD, NPPNH2, YWK, GAM, YPD, PVAR, IFAULT).

Formal parameters

NP	Integer	input: order of AR model
ALPHA	Real Array (NP)	input: coefficients of AR model
SIGSQ	Real	input: variance of white noise
Y	Real Array (NT2)	input: data vector

IOPT	Integer	input: option switch equal to 1 if both predictors and variances to be calculated 0 if only predictors desired
NT1	Integer	input: t_1 (first memory)
NT2	Integer	input: t_2 (last memory)
NH1	Integer	input: h_1 (first horizon)
NH2	Integer	input: h_2 (last horizon)
NYPD	Integer	input: (NT2-NT1+1)(NH2-NH1+1)
NPPNH2	Integer	input: NP + NH2
YWK	Real Array (NPPNH2)	workspace:
GAM	Real Array (NPPNH2)	output: $\gamma(1), \ldots, \gamma(\text{NP+NH2})$
YPD	Real Array (NYPD)	output: $((Y(t+h\|t), h = h_1, \ldots, h_2), t = t_1, \ldots, t_2)$
PVAR	Real Array (NH2)	output: if IOPT = 1, $\sigma^2_{t,h}$, $h = 1, \ldots, h_2$, if IOPT = 0, PVAR not used.
IFAULT	Integer	output: failure indicator

Failure Indications

Value of IFAULT	Meaning
0	no failure
1	NP < 1
2	NT1 < NP or NT1 > NT2
3	NH1 < 1 or NH1 > NH2
4	SIGSQ ≤ 0
5	IOPT not 0 or 1 or NYPD < (NT2-NT1+1)(NH2-NH1+1) or NPPNH2 < NP+NH2

SUBROUTINE MAPD (NQ, BETA, SIGSQ, Y, IOPT, NT1, NT2, NH1, NH2, NYPD, NPVAR, IROWS, DEL, RX, RX0, DX, ALX, MTWO, E, YPD, PVAR, IFAULT)

Formal parameters

NQ	Integer	input: order of MA model
BETA	Real Array (NQ)	input: coefficients of MA model
SIGSQ	Real	input: variance of white noise
Y	Real Array (NT2)	input: data vector
IOPT	Integer	input: option switch equal to: 1 if both predictors and variances to be calculated 0 if only predictors desired
NT1	Integer	input: t_1 (first memory)
NT2	Integer	input: t_2 (last memory)
NH1	Integer	input: h_1 (first horizon)
NH2	Integer	input: h_2 (last horizon)

NYPD	Integer	input: (NT2-NT1+1)(NH2-NH1+1)
NPVAR	Integer	input: Same as NYPD if IOPT = 1, $\geq$ 1 if IOPT = 0.
IROWS	Integer	input: row dimension of ALX and DX in calling program (see notes 6 and 8 above)
DEL	Real	input: absolute convergence criterion
RX	Real Array (NQ)	output: $R_X(1), \ldots, R_X(q)$
RXO	Real	output: $R_X(0)$
DX	Real Array (IROWS)	output: $[D_X]_i$, $i = 1, \ldots, M_2$
ALX	Real Array (IROWS,NQ)	output: $[L_X]_{ij}$, $i = 1, \ldots, M_2$, $j = i-q, \ldots, i-1$
MTWO	Integer	output: M_2 (see notes 6 and 8 above)
E	Real Array (NT2)	output: $e(1), \ldots, e(t_2)$
YPD	Real Array (NYPD)	output: $((Y(t+h\|t), h = h_1, \ldots, h_2), t = t_1, \ldots, t_2)$
PVAR	Real Array (NPVAR)	output: if IOPT = 1, $((\sigma^2_{t,h}), h = h_1, \ldots, h_2), t = t_1, \ldots, t_2)$, if IOPT = 0, PVAR not used
IFAULT	Integer	output: failure indicator

Failure Indications

Value of IFAULT	Meaning
0	no failure
1	NQ < 1 or IOPT not 0 or 1
2	NT1 < NQ+1 or NT1 > NT2
3	NH1 < 1 or NH1 > NH2
4	NYPD < (NT2-NT1+1)(NH2-NH1+1) or NPVAR < 1 or IOPT = 1 and NPVAR < (NT2-NT1+1)(NH2-NH1+1)
5	IROWS < 2
6	SIGSQ < 0
7	nonpositive $[D_X]_{ii}$ encountered
8	IROWS < NT2+NH2 and convergence not reached

Auxiliary algorithms

SUBROUTINE MACV (NQ, BETA, SIGSQ, RX, RXO, IFAULT)

Formal parameters

NQ	Integer	Input: order of MA model
BETA	Real Array (NQ)	input: coefficients of MA model
SIGSQ	Real	input: variance of white noise

RX	Real Array (NQ)	output: $R_X(1), \ldots, R_X(NQ)$
RXO	Real	output: $R_X(0)$
IFAULT	Integer	output: failure indicator equal to: 0 if no failure 1 if NQ < 1

SUBROUTINE MXCV (NP, NQ, M, IROWS, ALPHA, BETA, SIGSQ, RYE, RYEO, WKM, IP, RY, RYO, IFAULT)

Formal parameters

NP	Integer	input: order of AR part of model
NQ	Integer	input: order of MA part of model
M	Integer	input: highest lag to calculate (M $\geq$ max (NP,NQ))
IROWS	Integer	input: row dimension of WKM, IP in calling program (IROWS > max (NP,NQ))
ALPHA	Real Array (NP)	input: coefficients of AR part of model
BETA	Real Array (NQ)	input: coefficients of MA part of model
SIGSQ	Real	input: variance of white noise
RYE	Real Array (NQ)	output: $R_{Y\varepsilon}(-1), \ldots, R_{Y\varepsilon}(-NQ)$
RYEO	Real	output: $R_{Y\varepsilon}(0)$
WKM	Real Array (IROWS,IROWS)	workspace:
IP	Integer Array (IROWS)	workspace:
RY	Real Array (M)	output: $R_Y(1), \ldots, R_Y(M)$
RYO	Real	output: $R_Y(0)$
IFAULT	Integer	output: failure indicator

Failure Indications

Value of IFAULT	Meaning
0	no failure
1	NP < 1 or NQ < 1
2	M $\leq$ max (NP,NQ)
3	IROWS $\leq$ max (NP,NQ)
4	singular matrix encountered

The subroutines DECOMP and SOLV as described by Moler (1972) are called by subroutine MXCV.

RESTRICTIONS, TIME, STORAGE

If the zeros of $g(z)$ are not outside the unit circle, then one of the $\alpha_j(j)$ will be greater than or equal to one in magnitude thus giving IFAULT = 10 in MXPD. If the zeros of $h(z)$ are not outside the unit circle then an element of D_X will become nonpositive thus giving IFAULT = 12 in MXPD or IFAULT = 7 in MAPD.

The bulk of storage and computing time in MXPD is devoted to the $M_1 \times p$ matrix L_Z and the $M_2 \times q$ matrix L_X. The values M_1 and M_2 increase as the smallest zeros of $g(z)$ and $h(z)$ approach the unit circle.

REFERENCES

Moler, C.B. (1972). Algorithm 423. Linear equation solver. Comm. Ass. Comp. Mach., 274.

Newton, H.J. and Pagano, M. (1980). The finite memory prediction of covariance stationary time series. Submitted for publication.

Wilkinson, J.H. (1967). The solution of ill-conditioned linear equations", in Mathematical Methods for Digital Computers II, A. Ralston and H.S. Wilf, eds, 65-93.

```
      SUBROUTINE MXPD(NP,NQ,ALPHA,BETA,SIGSQ,Y,IOPT,NT1,NT2,NH1,NH2,
     1NYPD,NPVAR,NPPNH2,IROWS1,IROWS2,IROWS3,DEL,RYE,RYE0,RY,RY0,IP,
     1YWK,RZ0,RX,RX0,ALZI,ALZ,ALX1,ALX2,DX,MONE,MTWO,X,E,YPD,PVAR,
     1IFAULT)
C
C     THIS SUBROUTINE CALCULATES PREDICTORS YPD AND (OPTIONALLY)
C     PREDICTION VARIANCES PVAR FOR HORIZONS NH1,....,NH2 EACH FOR
C     MEMORIES NT1,....,NT2.
C
      DIMENSION ALPHA(NP),BETA(NQ),Y(NT2),RYE(NQ),RY(IROWS3),
     1IP(IROWS3),ALZI(IROWS3,IROWS3),ALZ(IROWS1,NP),YWK(NPPNH2),
     1ALX1(IROWS3,IROWS3),ALX2(IROWS2,NQ),DX(IROWS2),X(NT2),RX(NQ),
     1E(NT2),YPD(NYPD),PVAR(NPVAR)
      DATA ZERO,ONE,EPS/0.0,1.0,1.E-10/
C
C     CHECK INPUT PARAMETERS :
C
      IFAULT=1
      IF(NP.LT.1.OR.NQ.LT.1.OR.IOPT.LT.0.OR.IOPT.GT.1) GO TO 999
      IFAULT=2
      IF(NT1.LE.NP+NQ.OR.NT2.LT.NT1) GO TO 999
      IFAULT=3
      IF(NH1.LT.1.OR.NH2.LT.NH1) GO TO 999
      IFAULT=4
      NCK=(NT2-NT1+1)*(NH2-NH1+1)
      IF(NYPD.LT.NCK) GO TO 999
      IF(NPVAR.LT.1) GO TO 999
      IF(NPVAR.LT.NCK.AND.IOPT.EQ.1) GO TO 999
      IFAULT=5
      IF(NPPNH2.LT.NP+NH2) GO TO 999
      IF(IROWS1.LT.NP+2.AND.IOPT.EQ.1) GO TO 999
      IF(IROWS1.LT.1) GO TO 999
      IFAULT=6
      IF(IROWS2.LT.NP+NQ+1) GO TO 999
      IFAULT=7
      IF(IROWS3.LT.NP+NQ) GO TO 999
      IFAULT=8
      IF(SIGSQ.LE.ZERO) GO TO 999
C
C     FIND RY0,RY(1),....,RY(NP+NQ) :
C
      NPPNQ=NP+NQ
      CALL MXCV(NP,NQ,NPPNQ,IROWS3,ALPHA,BETA,SIGSQ,RYE,RYE0,
     1ALX1,IP,RY,RY0,IF1)
      IFAULT=9
      IF(IF1.EQ.4) GO TO 999
C
C     FIND ALZI,RZ0 (ALZI INITIALIZED, ALPHA(J,I) FORMED IN ALX1(J,I),
C     J.I.LE.NP, RZ0 FORMED, ALZI FORMED FROM ELEMENTS IN ALX1) :
C
      IFAULT=10
      DO 20 I=1,NPPNQ
      DO 10 J=1,NPPNQ
   10 ALZI(I,J)=ZERO
   20 ALZI(I,I)=ONE
      DO 30 I=1,NP
   30 ALX1(NP,I)=ALPHA(I)
C
      IF(NP.EQ.1) GO TO 50
      NPM1=NP-1
      DO 40 J=1,NPM1
```

```
      JJ=NPM1-J+1
      JJP1=JJ+1
      PART=ALX1(JJP1,JJP1)
      IF(PART.GE.ONE) GO TO 999
      DEN=ONE-PART*PART
         DO 40 I=1,JJ
         JJP1MI=JJP1-I
 40   ALX1(JJ,I)=(ALX1(JJP1,I)-PART*ALX1(JJP1,JJP1MI))/DEN
C
 50   RZ0=SIGSQ
      DO 60 J=1,NP
 60   RZ0=RZ0/(ONE-ALX1(J,J)*ALX1(J,J))
C
      IF(NP.EQ.1) GO TO 80
      DO 70 J=2,NP
      JM1=J-1
         DO 70 I=1,JM1
         JJ=JM1-I+1
 70   ALZI(J,I)=ALX1(JM1,JJ)
 80   CONTINUE
      NPP1=NP+1
      DO 90 I=NPP1,NPPNQ
      IFST=I-NPP1
         DO 90 J=1,NP
         JJ=IFST+J
         J1=NP-J+1
 90   ALZI(I,JJ)=ALPHA(J1)
C
C    IF(IOPT.EQ.1), FIND MONE AND ALZ (INVERT ALZI, FIND ELEMENTS
C    OF NP TH COLUMN OF ALZ   UNTIL CONVERGENCE,
C    THEN FILL IN REST OF ALZ) :
C
      IF(IOPT.EQ.0) GO TO 230
      DO 110 I=1,NP
      DO 100 J=1,NP
 100   ALZ(I,J)=ZERO
 110   ALZ(I,I)=ONE
      IF(NP.EQ.1) GO TO 140
      DO 130 J=2,NP
      JM1=J-1
         DO 130 K=1,JM1
         C=ZERO
         JMK=J-K
         JMKP1=JMK+1
            DO 120 IR=JMKP1,J
 120        C=C-ALZ(J,IR)*ALZI(IR,JMK)
 130  ALZ(J,JMK)=C
 140  CONTINUE
C
      CK=RZ0/SIGSQ
      NPP1=NP+1
      ALZ(NPP1,NP)=-ALPHA(1)
      SUMSQ=ONE+ALPHA(1)*ALPHA(1)
      MUP=MIN0(IROWS1-NP,NT2+NH2-NP)
      DO 160 J=2,MUP
      LLOW=MAX0(0,J-NP)+1
      LUP=J
      ALD=ZERO
      CC=ONE
         DO 150 LL=LLOW,LUP
         L=LL-1
```

```
         JML=J-L
         L1=NP+L
         IF(L.GT.0) CC=ALZ(L1,NP)
 150     ALD=ALD-ALPHA(JML)*CC
      NPPJ=NP+J
      ALZ(NPPJ,NP)=ALD
      SUMSQ=SUMSQ+ALD*ALD
      IF(ABS(SUMSQ-CK).LT.DEL) GO TO 180
 160  CONTINUE
      IF(MUP.LT.NT2+NH2-NP) GO TO 170
      MONE=NT2+NH2
      GO TO 190
 170  CONTINUE
      IFAULT=11
      MONE=IROWS1
      GO TO 999
 180  MONE=J+NP
 190  CONTINUE
C
      IF(NP.EQ.1) GO TO 220
      NPM1=NP-1
      M1MNP=MONE-NP
      DO 210 J=1,M1MNP
      NPPJ=NP+J
          DO 210 L=1,NPM1
          C=ZERO
              DO 200 IR=1,NP
              NPPJMR=NPPJ-IR
 200          C=C-ALPHA(IR)*ALZ(NPPJMR,L)
 210  ALZ(NPPJ,L)=C
 220  CONTINUE
C
C    FIND RX0,RX(1),....,RX(NQ) :
C
      CALL MACV(NQ,BETA,SIGSQ,RX,RX0,IF1)
C
C    FIND ALX1,DX(1),....,DX(NP+NQ) (NOTE THAT ALPHA(I,J) FOR
C    1.LE.J.LE.I.LT.NP IS IN ALZ1(I+1,I-J+1)) :
C
 230  CONTINUE
      DO 250 I=1,NPPNQ
      DO 240 J=1,NPPNQ
 240  ALX1(I,J)=ZERO
 250  ALX1(I,I)=ONE
      IFAULT=12
C
      DX(1)=RY0
      DO 340 I=2,NPPNQ
      IM1=I-1
          DO 290 J=1,IM1
          C=ZERO
              DO 260 L=1,I
              DO 260 M=1,J
              ILMM=IABS(L-M)
              IF(ILMM.EQ.0) C=C+ALZ1(I,L)*ALZ1(J,M)*RY0
              IF(ILMM.GT.0) C=C+ALZ1(I,L)*ALZ1(J,M)*RY(ILMM)
 260          CONTINUE
          IF(J.EQ.1) GO TO 280
          JM1=J-1
              DO 270 L=1,JM1
 270          C=C-ALX1(I,L)*DX(L)*ALX1(J,L)
```

```
 280       ALX1(I,J)=C/DX(J)
 290       CONTINUE
      C=RX0
      IF(I.GT.NP) GO TO 320
      C=ZERO
          DO 310 M=1,I
          C1=ALZI(I,M)
          C2=ZERO
              DO 300 L=1,I
              ILMM=IABS(M-L)
              IF(ILMM.EQ.0) C2=C2+ALZI(I,L)*RY0
              IF(ILMM.GT.0) C2=C2+ALZI(I,L)*RY(ILMM)
 300          CONTINUE
 310      C=C+C1*C2
 320      DO 330 L=1,IM1
 330      C=C-DX(L)*ALX1(I,L)*ALX1(I,L)
      IF(C.LT.EPS) GO TO 999
 340  DX(I)=C
C
C   FIND MTWO, ALX2, AND THE REST OF DX (IN THIS SECTION,
C   I AND J=I-NQ,....,I-1 REPRESENT THE INDICES OF THE NONZERO,
C   NONONE ELEMENTS OF THE MATRIX LSUBX. NOTE THAT IF INTEGERS
C   M.LE.N.LE.NP+NQ, THEN LSUBX(N,M) IS STORED IN LSUBX(N,M)
C   WHILE IF N.GT.NP+NQ, THEN LSUBX(N,M) IS IN ALX2(N-NP-NQ,
C   NQ-(N-M)+1), M=N-NQ,....,N-1) :
C
      IFAULT=12
      IUP=MIN0(IROWS2-NP-NQ,NT2+NH2-NP-NQ)
      DO 400 II=1,IUP
      I=NPPNQ+II
      IMNQ=I-NQ
      ALX2(II,1)=RX(NQ)/DX(IMNQ)
      IF(NQ.EQ.1) GO TO 370
          DO 360 JJ=2,NQ
          J=IMNQ+JJ-1
          NQ1=NQ-JJ+1
          J1=J-NPPNQ
          C=RX(NQ1)
          JJM1=JJ-1
              DO 350 LL=1,JJM1
              L=IMNQ+LL-1
              J2=NQ-(J-L)+1
              IF(J.LE.NPPNQ) C1=ALX1(J,L)
              IF(J.GT.NPPNQ) C1=ALX2(J1,J2)
 350          C=C-ALX2(II,LL)*DX(L)*C1
 360      ALX2(II,JJ)=C/DX(J)
 370  C=RX0
          DO 380 L=1,NQ
          LL=I-NQ+L-1
 380      C=C-ALX2(II,L)*ALX2(II,L)*DX(LL)
      DX(I)=C
      IF(DX(I).LT.EPS) GO TO 999
C
          DO 390 JJ=1,NQ
          NQMJP1=NQ-JJ+1
 390      IF(ABS(ALX2(II,JJ)-BETA(NQMJP1)).GT.DEL) GO TO 400
      IF(ABS(DX(I)-SIGSQ).GT.DEL) GO TO 400
      MTWO=II+NPPNQ
      GO TO 420
 400  CONTINUE
      IF(IUP.LT.NT2+NH2-NPPNQ) GO TO 410
```

```
      MTWO=NT2+NH2
      GO TO 420
  410 CONTINUE
      IFAULT=13
      MTWO=IROWS2-NPPNQ
      GO TO 999
  420 CONTINUE
C
C   FIND X(1).....X(NT2).E(1).....E(NT2) :
C
      NPP1=NP+1
      X(1)=Y(1)
      IF(NP.EQ.1) GO TO 450
      DO 440 J=2,NP
      C=Y(J)
      JM1=J-1
          DO 430 L=1,JM1
          JML=J-L
  430     C=C+ALZ1(J,JML)*Y(JML)
  440 X(J)=C
  450 DO 470 J=NPP1,NT2
      C=Y(J)
          DO 460 L=1,NP
          JML=J-L
  460     C=C+ALPHA(L)*Y(JML)
  470 X(J)=C
C
      E(1)=X(1)
      DO 490 J=2,NPPNQ
      C=X(J)
      JM1=J-1
          DO 480 L=1,JM1
  480     C=C-ALX1(J,L)*E(L)
  490 E(J)=C
      MLOW=NPPNQ+1
      MUP=MIN0(NT2,MTWO)
      DO 510 J=MLOW,MUP
      JJ=J-NPPNQ
      C=X(J)
          DO 500 L=1,NQ
          LL=J-NQ+L-1
  500     C=C-ALX2(JJ,L)*E(LL)
  510 E(J)=C
      IF(MUP.EQ.NT2) GO TO 540
      MUPP1=MUP+1
      DO 530 J=MUPP1,NT2
      C=X(J)
          DO 520 L=1,NQ
          JML=J-L
  520     C=C-BETA(L)*E(JML)
  530 E(J)=C
  540 CONTINUE
C
C   FIND YPD :
C
      NPDPT=NH2-NH1+1
      DO 630 NT=NT1,NT2
      NSOFAR=(NT-NT1)*NPDPT
      NTMNP=NT-NP
          DO 550 I=1,NP
          II=NTMNP+I
```

```
 550      YWK(I)=Y(II)
          DO 610 NH=1,NH2
          NPPNH=NP+NH
          NTPNH=NT+NH
          IROWLX=NTPNH-NPPNQ
          XTPH=ZERO
          IF(NH.GT.NQ) GO TO 590
          IF(NTPNH.GT.MTWO) GO TO 570
             DO 560 K=NH,NQ
             INDL=NQ-K+1
             INDE=NTPNH-K
 560         XTPH=XTPH+ALX2(IROWLX,INDL)*E(INDE)
          GO TO 590
 570         DO 580 K=NH,NQ
             INDE=NTPNH-K
 580         XTPH=XTPH+BETA(K)*E(INDE)
 590       C=XTPH
             DO 600 J=1,NP
             INDY=NPPNH-J
 600         C=C-ALPHA(J)*YWK(INDY)
 610       YWK(NPPNH)=C
           DO 620 I=NH1,NH2
           N10=NP+I
           N11=I-NH1+1
           N12=NSOFAR+N11
 620       YPD(N12)=YWK(N10)
 630    CONTINUE
C
C     IF IOPT.EQ.1, FIND PVAR :
C
        IF(IOPT.EQ.0) GO TO 690
        M1MNP=MONE-NP
        DO 680 NT=NT1,NT2
        NSOFAR=(NT-NT1)*NPDPT
            DO 680 NH=NH1,NH2
            NINDX=NSOFAR+NH-NH1+1
            NTPNH=NT+NH
            C=ZERO
               DO 670 KP1=1,NH
               K=KP1-1
               KROW=NTPNH-K
               C1=SIGSQ
               IF(KROW.LT.MTWO) C1=DX(KROW)
               C2=ONE
               IF(K.EQ.0) GO TO 670
               NPPK=NP+K
               C2=ALZ(NPPK,NP)
                  DO 660 IR=1,K
                  KMR=K-IR
                  C3=ZERO
                  IF(KMR.GT.M1MNP) GO TO 640
                  INDL=NP+KMR
                  C3=ALZ(INDL,NP)
 640              C4=ZERO
                  IF(IR.GT.NQ) GO TO 660
                  N10=NTPNH-KMR
                  IF(N10.GT.MTWO) GO TO 650
                  N11=N10-NPPNQ
                  N12=NQ-IR+1
                  C4=ALX2(N11,N12)
                  GO TO 660
```

```
 650             C4=BETA(IR)
 660             C2=C2+C3*C4
 670          C=C+C1*C2*C2
 680  PVAR(NINDX)=C
C
 690  CONTINUE
C
      IFAULT=0
 999  RETURN
      END
      SUBROUTINE MAPD(NQ,BETA,SIGSQ,Y,IOPT,NT1,NT2,NH1,NH2,
     1NYPD,NPVAR,IROWS,DEL,RX,RX0,DX,ALX,MTWO,E,YPD,PVAR,IFAULT)
C
C   THIS SUBROUTINE CALCULATES PREDICTORS YPD AND
C   (OPTIONALLY) PREDICTION VARIANCES PVAR FOR HORIZINS
C   NH1,....,NH2 EACH FOR MEMORIES NT1,....,NT2.
C
      DIMENSION BETA(NQ),Y(NT2),RX(NQ),DX(IROWS),ALX(IROWS,NQ),
     1E(NT2),YPD(NYPD),PVAR(NPVAR)
      DATA ZERO,ONE,EPS/0.0,1.0,1.E-10/
C
      IFAULT=1
      IF(NQ.LT.1.OR.IOPT.LT.0.OR.IOPT.GT.1) GO TO 199
      IFAULT=2
      IF(NT1.LT.NQ+1.OR.NT1.GT.NT2) GO TO 199
      IFAULT=3
      IF(NH1.LT.1.OR.NH1.GT.NH2) GO TO 199
      IFAULT=4
      NCK=(NT2-NT1+1)*(NH2-NH1+1)
      IF(NYPD.LT.NCK) GO TO 199
      IF(NPVAR.LT.1) GO TO 199
      IF(NPVAR.LT.NCK.AND.IOPT.EQ.1) GO TO 199
      IFAULT=5
      IF(IROWS.LT.2) GO TO 199
      IFAULT=6
      IF(SIGSQ.LE.ZERO) GO TO 199
      IFAULT=7
C
C   FIND MTWO AND ROWS OF ALX, DX :
C
      CALL MACV(NQ,BETA,SIGSQ,RX,RX0,IF1)
C
      DX(1)=RX0
      DO 10 I=1,NQ
  10  ALX(I,I)=ONE
      E(1)=Y(1)
      IUP=MIN0(IROWS,NT2+NH2)
      DO 100 I=2,IUP
      I1=MAX0(I-NQ-1,0)+1
      IM1=I-1
      NELTS=IM1-I1+1
          DO 30 J=I1,IM1
          JIND=J-I1+1
          IMJ=I-J
          JM1=J-1
          C=RX(IMJ)
          IF(J.EQ.I1) GO TO 30
          J1=MAX0(J-NQ-1,0)+1
              DO 20 L=I1,JM1
              LL=L-I1+1
              JJ=L-J1+1
```

```
   20          C=C-ALX(I,LL)*DX(L)*ALX(J,JJ)
   30       ALX(I,JIND)=C/DX(J)
         C=RX0
            DO 40 J=I1,IM1
            JIND=J-I1+1
   40       C=C-DX(J)*ALX(I,JIND)*ALX(I,JIND)
         IF(C.LE.EPS) GO TO 199
         DX(I)=C
         IF(I.GT.NT2) GO TO 60
         C=Y(I)
            DO 50 J=1,NELTS
            II=I-NELTS+J-1
   50       C=C-ALX(I,J)*E(II)
         E(I)=C
         IF(I.LE.NQ) GO TO 100
   60       DO 70 J=1,NQ
            JJ=NQ-J+1
   70       IF(ABS(ALX(I,J)-BETA(JJ)).GE.DEL) GO TO 100
         IF(ABS(DX(I)-SIGSQ).GE.DEL) GO TO 100
         MTWO=I
         IF(I.GE.NT2) GO TO 110
         IP1=I+1
            DO 90 J=IP1,NT2
            C=Y(J)
               DO 80 K=1,NQ
               JMK=J-K
   80          C=C-BETA(K)*E(JMK)
   90       E(J)=C
         GO TO 120
  100    CONTINUE
         IFAULT=8
         IF(IUP.LT.NT2+NH2) GO TO 199
  110    MTWO=NT2+NH2
  120    IFAULT=0
C
C     CALCULATE PREDICTORS :
C
         NPDPT=NH2-NH1+1
         DO 140 NT=NT1,NT2
         NSOFAR=(NT-NT1)*NPDPT
            DO 140 NH=NH1,NH2
            NIND=NSOFAR+NH-NH1+1
            YPD(NIND)=ZERO
            IF(NH.GT.NQ) GO TO 140
            NTPNH=NT+NH
            C=ZERO
               DO 130 K=NH,NQ
               INDE=NTPNH-K
               INDL=NQ-K+1
               C1=BETA(K)
               IF(NTPNH.LE.MTWO) C1=ALX(NTPNH,INDL)
  130          C=C+C1*E(INDE)
  140    YPD(NIND)=C
C
C     IF IOPT=1, CALCULATE VARIANCES :
C
         IF(IOPT.EQ.0) GO TO 199
         DO 180 NT=NT1,NT2
         NSOFAR=(NT-NT1)*NPDPT
            DO 180 NH=NH1,NH2
            NHM1=NH-1
```

```
      NIND=NSOFAR+NH-NH1+1
      NTPNH=NT+NH
      C=SIGSQ
      IF(NT.LT.MTWO) C=DX(NTPNH)
      IF(NH.EQ.1) GO TO 180
      NHUP=MINO(NHM1,NQ)
      IF(NT.GE.MTWO) GO TO 160
         DO 150 K=1,NHUP
         INDD=NTPNH-K
         INDL=NQ-K+1
150      C=C+DX(INDD)*ALX(NTPNH,INDL)*ALX(NTPNH,INDL)
      GO TO 180
160      DO 170 K=1,NHUP
170      C=C+SIGSQ*BETA(K)*BETA(K)
180   PVAR(NIND)=C
C
199   RETURN
      END
      SUBROUTINE ARPD(NP,ALPHA,SIGSQ,Y,IOPT,NT1,NT2,NH1,NH2,
     1NYPD,NPPNH2,YWK,GAM,YPD,PVAR,IFAULT)
C
C   THIS SUBROUTINE CALCULATES PREDICTORS YPD AND (OPTIONALLY)
C   PREDICTION VARIANCES PVAR FOR HORIZONS NH1,....NH2 EACH
C   FOR MEMORIES NT1,....NT2
C
      DIMENSION Y(NT2),ALPHA(NP),YWK(NPPNH2),GAM(NPPNH2),
     1YPD(NYPD),PVAR(NH2)
      DATA ZERO,ONE/0.0,1.0/
C
      IFAULT=1
      IF(NP.LT.1) GO TO 100
      IFAULT=2
      IF(NT1.LE.NP.OR.NT1.GT.NT2) GO TO 100
      IFAULT=3
      IF(NH1.LT.1.OR.NH1.GT.NH2) GO TO 100
      IFAULT=4
      IF(SIGSQ.LE.ZERO) GO TO 100
      IFAULT=5
      IF(IOPT.LT.0.OR.IOPT.GT.1) GO TO 100
      IFAULT=6
      NCK=(NT2-NT1+1)*(NH2-NH1+1)
      IF(NYPD.LT.NCK.OR.NPPNH2.LT.NP+NH2) GO TO 100
      IFAULT=0
C
C   FIND PREDICTIONS :
C
      NPDPT=NH2-NH1+1
      DO 50 NT=NT1,NT2
      NSOFAR=(NT-NT1)*NPDPT
      NTMNP=NT-NP
         DO 10 I=1,NP
         II=NT-NP+I
10       YWK(I)=Y(II)
         DO 30 NH=1,NH2
         NPPNH=NP+NH
         NTPNH=NT+NH
         C=ZERO
            DO 20 I=1,NP
            II=NPPNH-I
20          C=C-ALPHA(I)*YWK(II)
30       YWK(NPPNH)=C
```

```
          DO 40 NH=NH1,NH2
          INDNH=NSOFAR+NH-NH1+1
          INDWK=NP+NH
   40     YPO(INDNH)=YWK(INDWK)
   50  CONTINUE
C
C    IF IOPT=1, FIND VARIANCES :
C
       IF(IOPT.EQ.0) GO TO 100
       GAM(1)=-ALPHA(1)
       IF(NH2.EQ.1) GO TO 80
       DO 70 NH=2,NH2
       LLOW=MAX0(0,NH-NP)+1
       C=ZERO
          DO 60 LL=LLOW,NH
          L=LL-1
          C1=ONE
          IF(L.GT.0) C1=GAM(L)
          NHML=NH-L
   60     C=C-ALPHA(NHML)*C1
   70  GAM(NH)=C
   80  PVAR(1)=SIGSQ
       IF(NH2.EQ.1) GO TO 100
       DO 90 I=2,NH2
       IM1=I-1
   90  PVAR(I)=PVAR(IM1)+SIGSQ*GAM(IM1)*GAM(IM1)
  100  RETURN
       END
       SUBROUTINE MACV(NQ,BETA,SIGSQ,RY,RYO,IFAULT)
C
C    THIS SUBROUTINE CALCULATES MA(NQ) AUTOCOVARIANCES OF LAGS
C    0,....,NQ (NQ.GT.0)
C
       DIMENSION BETA(NQ),RY(NQ)
       DATA ONE/1.0/
C
       IFAULT=1
       IF(NQ.LT.1) GO TO 40
       IFAULT=0
C
       C=ONE
       DO 10 I=1,NQ
   10  C=C+BETA(I)*BETA(I)
       RYO=C*SIGSQ
C
       DO 30 IV=1,NQ
       C=BETA(IV)
       IF(IV.EQ.NQ) GO TO 30
       NQMIV=NQ-IV
          DO 20 J=1,NQMIV
          JPIV=J+IV
   20     C=C+BETA(J)*BETA(JPIV)
   30  RY(IV)=C*SIGSQ
   40  RETURN
       END
       SUBROUTINE MXCV(NP,NQ,M,IROWS,ALPHA,BETA,SIGSQ,RYE,RYEO,
      1WKM,IP,RY,RYO,IFAULT)
C
C    THIS SUBROUTINE CALCULATES ARMA(NP,NQ) AUTOCOVARIANCES FOR
C    LAGS 0,....,M (M.GE.MAX(NP,NQ), NP,NQ.GT.0) :
C
```

```
      DIMENSION ALPHA(NP),BETA(NQ),RYE(NQ),WKM(IROWS,IROWS),
     1IP(IROWS),RY(M)
      DATA ONE,ZERO/1.0,0.0/
C
      IFAULT=1
      IF(NQ.LT.1.OR.NP.LT.1) GO TO 110
      IFAULT=2
      MAXPQ=MAX0(NP,NQ)
      MM=MAXPQ+1
      IF(M.LE.MAXPQ) GO TO 110
      IFAULT=3
      IF(IROWS.LT.MM) GO TO 110
      IFAULT=4
C
C    FIND RYE0,RYE(1),....,RYE(NQ) :
C
      RYE0=SIGSQ
      DO 30 IV=1,NQ
      C=SIGSQ*BETA(IV)
      NUP=MIN0(IV,NP)
         DO 20 J=1,NUP
         IVMJ=IV-J
         IF(IVMJ.EQ.0) GO TO 10
         C=C-ALPHA(J)*RYE(IVMJ)
         GO TO 20
   10    C=C-ALPHA(J)*RYE0
   20    CONTINUE
   30 RYE(IV)=C
C
C    USE DECOMP, SOLV TO OBTAIN RY0,RY(1),....,RY(MAX(NP,NQ)) :
C
      DO 40 IV=1,MM
      RY(IV)=ZERO
         DO 40 J=1,MM
   40 WKM(IV,J)=ZERO
C
      NPP1=NP+1
      NQP1=NQ+1
      DO 60 IVP1=1,NQP1
      IV=IVP1-1
      C=RYE0
      IF(IV.GT.0) C=C*BETA(IV)
      IF(IV.EQ.NQ) GO TO 60
         DO 50 K=IVP1,NQ
         KMIV=K-IV
   50    C=C+BETA(K)*RYE(KMIV)
   60 RY(IVP1)=C
C
      DO 70 IVP1=1,MM
      IV=IVP1-1
      WKM(IVP1,IVP1)=WKM(IVP1,IVP1)+ONE
         DO 70 J=1,NP
         I1=IABS(IV-J)+1
   70 WKM(IVP1,I1)=WKM(IVP1,I1)+ALPHA(J)
C
      CALL DECOMP(MM,IROWS,WKM,IP)
      IF(IP(MM).EQ.0) GO TO 110
      IFAULT=0
      CALL SOLV(MM,IROWS,WKM,RY,IP)
      RY0=RY(1)
      DO 80 IV=1,MAXPQ
```

```
      IVP1=IV+1
   80 RY(IV)=RY(IVP1)
C
C   USE DIFFERENCE EQUATION TO GET THE REST OF THE RY :
C
      IF(M.EQ.MAXPQ) GO TO 110
      DO 100 IV=MM,M
      C=ZERO
         DO 90 J=1,NP
         IVMJ=IV-J
   90    C=C-ALPHA(J)*RY(IVMJ)
  100 RY(IV)=C
C
  110 RETURN
      END
```

END

DATE
FILMED
10-81

DTIC